Mateusz Ziomek

22.04.2018r

**Ogólne zasady działania**

Czas odświerzania (iteracja) jest stały i ustawoiny ja 0.1s (delay(100)) tego nie zmieniamy (powinno być wystarczająco szybko dla piłki na najszybszy poziom), jeżeli chcemy zmienić szybkość poruszania czegoś to bawimy się z licznikami (xCount, yCount, switchCount). Działają one tak że liczą przez ile iteracji była wykonywana czynność i dopiero jak osiągną pewną wartość wykonują tą akcję (czyli np. gdy JoyStick trzymany do góry przez 0.3s czyli 3 iteracje, wartość yCount = 3, wtedy paletka idzie do góry i zeruje licznik jak w dół to ujemne wartości). Przykład:

if(yCount >= 3){ //licznik osiągnął wartość 3 yCount = 0; //zeruje go if(paddDotPos < 8){ //sprawdza czy paletka nie wyszła za ekran ++paddDotPos; //jeśli nie to przesówa convPaddleToSize(); //konwertuje pozycje do tablicy } Serial.println("Down");//wypisuje informacje o akcji w monitorze

cała opisana logika jest zrobiona w funkcji checkCounters() włączne z zakomentowanymi operacjami dla xCounter i krótkiego przyciśnięcia.

Przesówanie paletki odbywa się w 2 etapach, zmiana pozycji indexu głównego pozycji paletki (int) paddDotPos i konwersja go do tablicy do wyświetlenia.

++paddDotPos; convPaddleToSize();

**Paddle2.ino** – moja część programu

Przesówanie paletki góra dół, paletka może się w całości schować (na górze lub na dole). rSize odpowiada za wielkość paletki, która zmienia się cyklicznie przy dłuższym przytrzymaniu przycisku.

Opis podłączenia do arduino (ważne żeby pamiętać że X\_pin i Y\_pin jest podłączone do portu A0 i A1 , a nie do portu 0, 1):

//JoyStick conectionconst int SW\_pin = 2; // digital pin connected to switch output

const int X\_pin = 0; // analog pin connected to X output podłączyć do !!! A0 !!!

const int Y\_pin = 1; // analog pin connected to Y output podłączyć do !!! A1 !!!

//8x8 Led conection

int DIN = 5; //DIN port 5

int CS = 4; //CS port 4

int CLK = 3; //CLK port 3

LedControl lc = LedControl(DIN,CLK,CS,0); //LED setup

**Zmienne globalne:**

byte paddlePos[8] = {0,0,0,0,0,0,0,0} - tablica przedstawiająca paletkę tworzona za pomocą paddDotPos i rSize, służy tylko do wyświetlania ewentualnie sprawdzania odbić

int paddDotPos = 2 - pozycja głównego punktu paletki na podstawie którego   
w połączeniu z rSize tworzona jest tablica paddlePos, przyjmuje wartości od -5 do 8

int rSize = 3 - wielkość paletki, działa w połączeniu z paddDotPos i tworzy tablicę paletki przyjmuje wartości od 1 do 5

int xCount = 0 - licznik zmian dla osi X int yCount = 0 - licznik zmian dla osi Y int switchCount = 0 - licznik zmian dla przycisku

int tmpPrntCount = 0 - zmienna stworzona dla celów testowych żeby wyświetlać rzeczy na monitorze bo wyświetlacz nie zawsze współpracuje można usunąć

**void setup()**

Tam są funkcje wykonywane na początku programu. Ustawienie ekranu i JoySticka. Z mojej strony dodana funkcja convPaddleToSize() żeby wypełnić tablicę paddlePos bo na start jest pusta.

**void loop()**

void loop() { ++tmpPrntCount; checkCounters(); if(tmpPrntCount >= 5){ //tego ifa można usunąc, jest tylko żeby testować tmpPrntCount = 0; Serial.println(rSize); printArray(paddlePos); }

// lc.clearDisplay(0); //czyści ekran printByte(paddlePos); delay(100);}

++tmpPrntCount – w celach testowych żeby cyklicznie co 5 iteracji wyświetliło tablicę, można usunąć.

checkCounters() - sprawdza czy jakieś akcje na JoySticku zostały wykonane, więcej   
w opisie funkcji.

if(tmpPrntCount >= 5){...} - w celach testowych, wyświetla tablicę paletki   
i wielkość co 5 iteracji, można usunąć.

printByte(paddlePos) - wyświetla tablicę z paletką na ekranie, o ile ekran akurat współpracuje.

delay(100) - czeka na kolejną iterację, iteracja co 0.1s (wydaje mi się że to wystarczająca szybkość maksymalna) lepiej tego nie zmieniamy i nie dodawać więcej takich funkcji, spowalnianiem   
i przyspieszaniem akcji operować na licznikach, jak to się zmieni to wszystko się zepsuje.

**void printArray(byte arr[])**

Funkcja odpowiedzialna za wyświetlanie podanej tablicy w monitorze, prosta w ogarnięciu więc nie będę tłumaczył.

**void printByte(byte character[])**

funkcja odpowiedzialna za wyświetlanie podanej tablicy na wyświetlaczu, prosta w ogarnięciu więc nie będę tłumaczył.

**void checkCounters()**

Funkcja odpowiedzialna z a sprawdzanie czy wystąpiła jakaś akcja, za pomocą liczników steruje się po jakim czasie ma wciśnięcia ma wykonać się akcja.

Zmienne odczytujące wartości z JoySticka:

int xRead = analogRead(X\_pin); int yRead = analogRead(Y\_pin); int switchRead = digitalRead(SW\_pin);

Wartości odczytów znajdują się pomiędzy 0 a 1023. Raczej oczywiste działanie. Zwrócić uwagę że jest analogRead i digitalRead.

Funkcja sprawdzająca czy występuje jakaś akcja, jeśli tak to zwiększająca/zmniejszająca (góra to wychylenie ponad 700, dół poniżej 300, tak przyjąłem powinno być dobrze wartości wychyleń odczytywane z **yRead**) licznik w zależności od akcji, jeżeli akcja przestaje być wykonywana to licznik jest zerowany, na przykładzie yCount:

if(yRead > 700 && yCount >= 0){ ++yCount; }else if(yRead < 300 && yCount <= 0){ --yCount; }else if(yRead >= 300 && yRead <= 700 & yCount != 0){ yCount = 0; }

Dla przycisku są 2 stany 0 – przyciśnięty, 1 – zwolniony. Funkcja analogiczna:

if(switchRead == 0){ ++switchCount; }else if(switchRead == 1 & switchCount != 0){ switchCount = 0; }

Jeżeli akcja wykonywana byłą przez odpowiednią ilość iteracji (w tym przypadku 3, 0.3s) to następuje zmiana pozycji w tym przypadku paletki. Czyli wyzerowanie licznika, dodanie/odjęcie do paddDotPos jedynki konwersja zmiany do tablicy paletki paddlePos i wyświetlenie informacji   
o akcji w monitorze.

if(yCount >= 3){ yCount = 0; if(paddDotPos < 8 ++paddDotPos; convPaddleToSize(); } Serial.println("Down"); }

Akcja dla przycisku dzieli się na krótkie i długie przyciśnięcie, jeżeli trzymamy przycisk cały czas to krótkie przyciśnięcie wykona się **tylko raz** po 3 iteracjach, a długie będzie się wykonywało cyklicznie co 10 iteracji, można zmienić.

Podczas długiego przyciśnięcia zmienia się wielkość paletki rSize i w odpowiedni sposób zmienia tablicę paletki paddlePos.

**convPaddleToSize()**

Na podstawie paddDotPos i rSize tworzy tablicę paletki paddlePos.

**Paletka1.ino** – inny program który może się przydać do przesówania punktów

Program przesówa punkt w zdefiniowany wcześniej sposób w funkcji loop().

Funkcja direction(int) na podstawie liczby wybiera gdzie ma zostać przesunięty punkt.

Funkcje goUp(), goDown(), goLeft(), goRight() realizują przesunięcia w zadanym kierunku.

Funkcja posOfIndex() znajduje i zwraca indeks pierwszego wystąpienia liczby innej niż 0, czyli w tym przypadku indeks punktu.